```
def min_length_after_removals(nums):
    n = len(nums)
    max_pairs = n // 2
    min_length = n - 2 * max_pairs
    return min_length
nums = [1, 2, 3, 4]
print(min_length_after_removals(nums))



def sub_str(words):
    result=[]
    for i in range(len(words)):
        for j in range(len(words)):
            if i!=j and words[i] in words[j]:
                result.append(words[i])
    return result
words=['has','as','deepika','deep']
print(sub_str(words))

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def sortedArrayToBST(nums):
    if not nums:
        return None
    def helper(left, right):
        if left > right:
            return None
```

```
        mid = (left + right) // 2
        root = TreeNode(nums[mid])
        root.left = helper(left, mid - 1)
        root.right = helper(mid + 1, right)
        return root
    return helper(0, len(nums) - 1)

def printLevelOrder(root):
    if not root:
        return []

    result = []
    queue = [root]

    while queue:
        current = queue.pop(0)
        if current:
            result.append(current.val)
            queue.append(current.left)
            queue.append(current.right)
        else:
            result.append(None)

    while result and result[-1] is None:
        result.pop()

    return result

nums = [-10, -3, 0, 5, 9]
tree_root = sortedArrayToBST(nums)
print(printLevelOrder(tree_root))
```

```
def wiggleSort(nums):
    nums.sort()
    n = len(nums)
    mid = (n + 1) // 2
    left = nums[:mid]
    right = nums[mid:]
    nums[::2], nums[1::2] = left[::-1], right[::-1]
nums1 = [1, 5, 1, 1, 6, 4]
wiggleSort(nums1)
print(nums1)
```